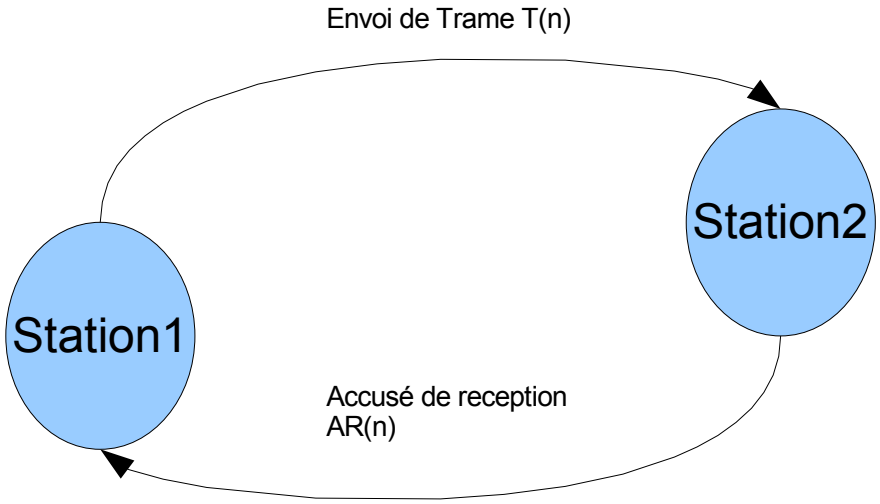
PROJET C : SIMULATION DE PROTOCOLE STOP& WAIT – ACK (sous Windows XP)

Implémentation du protocole Stop and Wait avec „fenêtre d'anticipation“ .



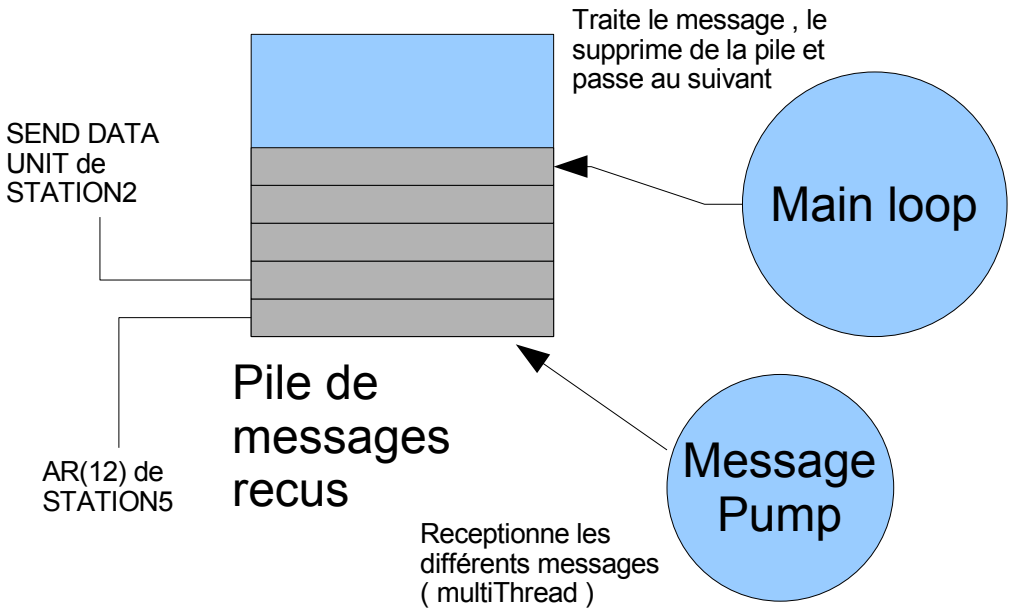
I. SANS FENETRE D'ANTICIPATION

- STATION1 communique avec STATION2 par le mécanisme suivant :
  - elle envoie une trame T(n) et attend un accusé de reception AR(n) de STATION2
  - Des reception de cet accusé ,STATION1 envoie la trame T(n+1)
- Si STATION2 ne renvoie pas un accusé de reception AR(n) dans un délai < TIMEOUT , STATION1 renvoie la trame AR(n) jusqu'à ce qu'elle recoive l'accusé de réception AR(n)

Pendant le temps où STATION1 attends l'accusé de reception AR(n) elle stoppe ses opérations avec STATION2 („STOP AND WAIT „) . A priori , cependant , rien ne l'empêche de traiter ses connexions avec d'autres stations .

STATION1 est donc susceptible de recevoir et d'envoyer des messages via d'autres stations STATION3, STATION4 , ..

Il est donc nécessaire d'une part d'implémenter un mécanisme qui permette à STATION1 de parcourir l'ensemble des messages qu'elle recoit et de les traiter . En l'absence d'une quelconque priorité , on traitera ces messages suivant le principe du „premier arrivé , premier servi“ (FIFO)



Comme l'attente d'un accusé de reception ne doit pas bloquer le traitement des autres opérations , un traitement multithread s'impose .

On créera donc autant de threads que de connexions en cours de traitement . Le programme principal devra donc s'assurer de la synchronization entre ces différentes thread afin d'éviter les deadlocks .

Ainsi , si nous admettons que les stations communiquent sur un port prédéfini

La „pompe à message“ de notre programme va réceptionner les évenements des divers threads de connexion et alimenter la pile de messages .

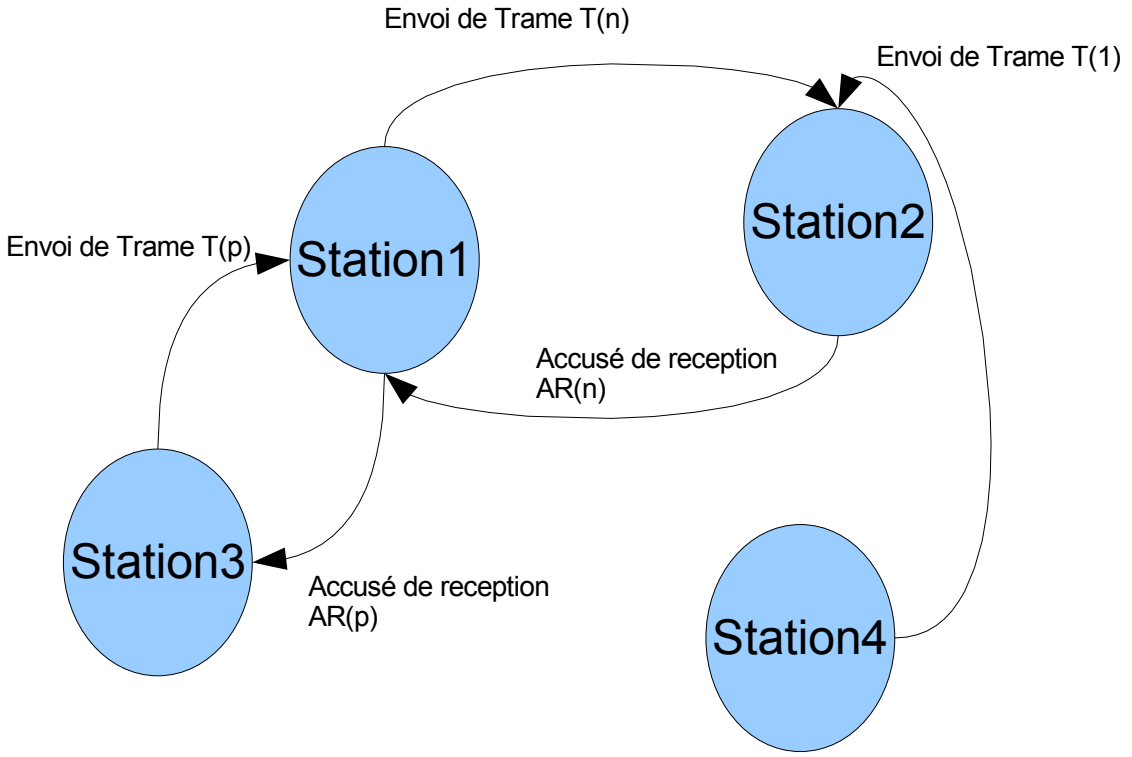
Un Watchdog doit faire sortir un thread de son état d'attente aprés la durée de TIMOUT et redémarrer le service d'envoi de trame au niveau de la thread .

Mode non connecté

On fonctionnera comme en mode connecté mais la connexion sera automatiquement acceptée avant le transfert et automatiquement détruite après le transfert de la data unit (il s'agit d'une simulation de protocole)

Etablissement de la connection pour le mode connecté

De même que pour le fonctionnement des serveurs de type FTP , on peut supposer que les stations recoivent des demandes de connexion sur un port fixe (exemple le port **2000**) et que dans le cas où elles acceptent ces connexions, elles allouent un port pour le transfert de données . Des que la connexion est acceptée , un nouveau thread de connexion est crée .



„Shell“ pour les stations

Afin de pouvoir lancer les scénarios , on developpera un mini-shell pour les stations correspondant aux différentes possibilités . L'entrée pourra se faire en ligne de commande ou via un fichier de commandes .

Ligne de commande : génerateur de scenarios

Le programme utilisera la console standard windows 32 bits .

Le but est de le présenter comme un programme de démonstration / Simulation du protocole STOP AND WAIT

Il presentera à l'utilisateur une suite de fenetres de type DOS permettant de :

- régler les différents paramètres (port d'écoute du serveur , probabilité d'erreur dans une trame réseau , timeout , niveau de verbosité , etc ...)
- choisir le nombre N de stations
- choisir des scénarios / tests
- lancer/stopper la démonstration / test

Le logiciel lancera alors les N stations et le scénario

On utilisera un générateur aléatoire sur le nombre de stations pour déterminer quelles stations émettent et à qui . On pourra éventuellement aussi répéter ces séquences .

Exemple d'un scénario généré :

- 4 stations
- probabilité d'envoi de trame avec erreur de 5%
- time out de 50 000 ms
- station1 effectue une demande de connexion a station 2 pour transférer un fichier „test.txt“
- station 1 transfere un fichier „test2.txt“ a station 3 sans connexion prélable
- station 2 effectue une demande de connexion a station 4 pour transférer un fichier „test.txt“
